

Improving the Randomization Step in Feasibility Pump

Santanu S. Dey^{*1}, Andres Iroume^{†1}, Marco Molinaro^{‡2}, and Domenico Salvagnin^{§3}

¹School of Industrial and Systems Engineering, Georgia Institute of Technology,
Atlanta, United States

²Computer Science Department, PUC-Rio, Brazil

³IBM Italy and DEI, University of Padova, Italy

September 27, 2016

Abstract

Feasibility pump (FP) is a successful primal heuristic for mixed-integer linear programs (MILP). The algorithm consists of three main components: rounding fractional solution to a mixed-integer one, projection of infeasible solutions to the LP relaxation, and a randomization step used when the algorithm stalls. While many generalizations and improvements to the original Feasibility Pump have been proposed, they mainly focus on the rounding and projection steps.

We start a more in-depth study of the randomization step in Feasibility Pump. For that, we propose a new randomization step based on the WalkSAT algorithm for solving SAT instances. First, we provide *theoretical analyses* that show the potential of this randomization step; to the best of our knowledge, this is the first time any theoretical analysis of running-time of Feasibility Pump or its variants has been conducted. Moreover, we also conduct computational experiments incorporating the proposed modification into a state-of-the-art Feasibility Pump code that reinforce the practical value of the new randomization step.

1 Introduction

Primal heuristics are used within mixed-integer linear programming (MILP) solvers for finding good integer feasible solutions quickly [FL11]. *Feasibility pump* (FP) is a very successful primal heuristic for mixed-binary LPs that was introduced in [FGL05]. At its core, Feasibility Pump is an *alternating projection method*, as described below.

Algorithm 1 Feasibility Pump (Naïve version)

- 1: **Input:** mixed-binary LP (with binary variables x and continuous variables y)
 - 2: Solve the linear programming relaxation, and let (\bar{x}, \bar{y}) be an optimal solution
 - 3: **while** \bar{x} is not integral **do**
 - 4: (Round) Round each coordinate of \bar{x} to the closest integer, call the obtained vector \tilde{x}
 - 5: (Project) Let (\bar{x}, \bar{y}) be the point in the LP relaxation that minimizes $\sum_i |x_i - \tilde{x}_i|$
 - 6: **end while**
 - 7: Return (\bar{x}, \bar{y})
-

The scheme presented above may *stall*, since the same infeasible integer point may be visited in Step 4 at different iterations. Whenever this happens, the paper [FGL05] recommends a *randomization step*, that after Step 4 flips the value of some of the binary variables as follows: Defining the *fractionality*

^{*}santanu.dey@isye.gatech.edu

[†]airoume3@gatech.edu

[‡]molinaro.marco@gmail.edu

[§]dominiqs@gmail.com

of variable x_i as $|\bar{x}_i - \tilde{x}_i|$ and let NN be the number of variables with positive fractionality, randomly generate a positive integer TT and flip $\min\{TT, NN\}$ variables with largest fractionality.

Together with a few other tweaks, this surprisingly simple method works very well. On MIPLIB 2003 instances, FP finds feasible solutions for 96.3% of the instances in reasonable time [FGL05].

Due to its success, many improvements and generalizations of FP (both for MILPs and mixed integer non-linear programs (MINLPs)) have been studied [AB07, BFL07, BCLM09, FS09, SLR10, DFLL10, BEET12, DFLL12, BEE⁺14]. However, the focus of these improvements has been on the projection and rounding steps or generalization for MINLPs; to the best of our knowledge, they use essentially the same randomization step as proposed in the original algorithm [FGL05] (and its generalization to the general integer MILP case of [BFL07]).

Moreover, even though FP is so successful and so many variants have been proposed, there is very limited theoretical analysis of its properties [BEET12]. In particular, to the best of our knowledge there is no known bounds on expected running-time of FP.

2 Our contributions

In this paper, we start a more in-depth study of the randomization step in Feasibility Pump. For that, we propose a new randomization step RANDWALKSAT_ℓ and provide both *theoretical analysis* as well as *computational experiments* in a state-of-the-art Feasibility Pump code that show the potential of this method.

Theoretical justification of RandWalkSAT_ℓ . The new randomization step RANDWALKSAT_ℓ is inspired by the classical algorithm WALKSAT [Sch99] for solving SAT instances (see also [Pap91, MJPL92]). The key idea of RANDWALKSAT_ℓ is that whenever Feasibility Pump stalls, namely an infeasible mixed-binary solution is revisited, it should flip a binary variable that participates in an *infeasible constraint*. More precisely, RANDWALKSAT_ℓ constructs a *minimal (projected) infeasibility certificate* for this solution and *randomly picks* a binary variable in it to be flipped (see Section 3 for exact definitions).

While the vague intuition that such randomization is trying to “fix” the infeasible constraint is clear, we go further and provide theoretical analyses that formally justify this and highlight more subtle advantageous properties of RANDWALKSAT_ℓ .

First, we analyze what happens if we simply repeatedly use *only* the new proposed randomization step RANDWALKSAT_ℓ , which gives a simple primal heuristic that we denote by MBWALKSAT . Not only we show that MBWALKSAT is guaranteed to find a solution if one exists, but its behavior is related to the (almost) *decomposability* and *sparsity* of the instance. To make this precise, consider a decomposable mixed-binary set with k blocks:

$$\begin{aligned} P^I &= P_1^I \times \dots \times P_k^I, \text{ where for all } i \in [k] \text{ we have} \\ P_i^I &= P_i \cap (\{0, 1\}^{n_i} \times \mathbb{R}^{d_i}), \quad P_i = \{(x^i, y^i) \in [0, 1]^{n_i} \times \mathbb{R}^{d_i} : A^i x^i + B^i y^i \leq b^i\}. \\ \text{Let } P &= P_1 \times \dots \times P_k \text{ denote the LP relaxation of } P^I. \end{aligned} \tag{1}$$

Note that since we allow $k = 1$, this also captures a general mixed-binary set. We then have the following running-time guarantee for the primal heuristic MBWALKSAT .

Theorem 2.1. *Consider a feasible decomposable mixed-binary set as in equation (1). Let s_i be such that each constraint in P_i^I has at most s_i binary variables, and define $c_i := \min\{s_i \cdot (d_i + 1), n_i\}$. Then with probability at least $1 - \delta$, MBWALKSAT with parameter $\ell = 1$ returns a feasible solution within $\ln(k/\delta) \sum_i n_i 2^{n_i \log c_i}$ iterations. In particular, this bound is at most $\bar{n}k 2^{\bar{n} \log \bar{n}} \cdot \ln(k/\delta)$, where $\bar{n} = \max_i n_i$.*

There are a few interesting features of this bound that indicates good properties of the proposed randomization step, apart from the fact that it is already able to find feasible solutions by itself. First, it depends on the *sparsity* s_i of the blocks, giving better running times on sparser problems. More importantly, the bound indicates that the algorithm works almost *independently* on each of the blocks, that is, it just takes about 2^{n_i} iterations to find a solution for each of the blocks, instead of $2^{n_1 + \dots + n_k}$ of a complete enumeration over the whole problem. In fact, the proof of Theorem 2.1 makes explicit this

almost independence of the algorithm over the blocks, and motivates the uses of *minimal* infeasibility certificates. Moreover, we note the important point that the algorithm is not provided the knowledge of the decomposability of the instance, it just *automatically* runs “fast” when the problem is decomposable. This gives some indication that the proposed randomization could still exhibit good behavior on the *almost decomposable* instances often found in practice (see discussion in [DMW16]).

RandWalkSAT $_\ell$ in conjunction with FP. Next, we analyze RANDWALKSAT $_\ell$ in the context of Feasibility Pump by adding it as a randomization step to the Naïve Feasibility Pump algorithm (Algorithm 1); we call the resulting algorithm WFP. This now requires understanding the complicated interplay of the randomization, rounding and projection steps: While in practice rounding and projection greatly help finding feasible solutions, their worst-case behavior is difficult to analyze and in fact they could take the iterates far away from feasible solutions. Although the general case is elusive at this point, we are nonetheless able to analyze the running time of WFP for *decomposable subset-sum* instances.

Definition 2.2. A separable subset-sum set is one of the form

$$\{(x^1, x^2, \dots, x^k) \in \{0, 1\}^{n_1+n_2+\dots+n_k} : a^i x^i = b_i \ \forall i\} \quad (2)$$

for non-negative (a^i, b_i) 's.

While this may seem like a simple class of problems, on these instances Feasibility Pump with the original randomization step from [FGL05] (without restarts) may not even converge, as illustrated next.

Remark 2.3. Consider the feasible subset-sum problem

$$\begin{aligned} \max \quad & x_2 \\ \text{s.t.} \quad & 3x_1 + x_2 = 3 \\ & x_1, x_2 \in \{0, 1\}. \end{aligned}$$

Consider the execution of the original Feasibility Pump algorithm (without restarts). The starting point is an optimal LP solution; without loss of generality, suppose it is the solution $(\frac{2}{3}, 1)$. This solution is then rounded to the point $(1, 1)$, which is infeasible. This point is then ℓ_1 -projected to the LP, giving back the point $(\frac{2}{3}, 1)$, which is then rounded again to $(1, 1)$. At this point the algorithm has stalled and applies the randomization step. Since only variable x_2 has strictly positive fractionality $|\frac{2}{3} - 1| = \frac{1}{3}$, only the first coordinate of $(1, 1)$ is a candidate to be flipped. So suppose this coordinate is flipped. The infeasible point $(0, 1)$ obtained is then ℓ_1 -projected to the LP, giving again the point $(\frac{2}{3}, 1)$. This sequence of iterates repeats indefinitely and the algorithm does not find the feasible solution $(1, 0)$.

The issue in this example is that the original randomization step never flips a variable with zero fractionality. Moreover, in Section B of the appendix we show that even if such flips are considered, there is a more complicated subset-sum instance where the algorithm stalls.

On the other hand, we show that algorithm WFP with the proposed randomization step always finds a feasible solution of feasible subset-sum instances, and moreover its running time again depends on the sparsity and the decomposability of the instance (in order to simplify the proof, we assume that $\tilde{x} \notin P$, then $\ell_1\text{-proj}(P, \tilde{x})$ is a vertex of P ; notice that since $\ell_1\text{-proj}(P, \tilde{x})$ is a linear programming problem and subset-sum instances are bounded, there is always a vertex satisfying the desired properties from $\ell_1\text{-proj}$).

Theorem 2.4. Consider a feasible separable subset-sum set P as in (2). Then with probability at least $1 - \delta$, WFP with $\ell = 2$ returns a feasible solution within $T = \lceil \ln(k/\delta) \rceil \sum_i n_i 2^{2n_i \log n_i} \leq \bar{n} k 2^{2\bar{n} \log \bar{n}}$. $\ln(k/\delta)$ iterations, where $\bar{n} = \max_i n_i$.

To the best of our knowledge this is the first theoretical analysis of the running-time of a variant of Feasibility Pump algorithm, even for a special class of instances. As in the case of repeatedly using just RANDWALKSAT $_\ell$, the algorithm WFP essentially works independently on each of the blocks (inequalities) of the problem, and has reduced running time on sparser instances.

The high-level idea of the proof Theorem 2.4 is to: 1) Show that the combination of projection plus rounding is *idempotent* for these instances, namely applying them once or repeatedly yields the same effect (Lemma 4.3); 2) Show that a round of randomization step plus projection plus rounding has a non-zero probability of generating an iterate closer to a feasible solution (Lemma 4.6).

Computational experiments. While the analyses above give insights on the usefulness of using RANDWALKSAT_ℓ in the randomization step of FP, in order to attest its practical value it is important to understand how it interacts with complex engineering components present in current Feasibility Pump codes. To this end, we considered the state-of-the-art code of [FS09] and modified its randomization step based on RANDWALKSAT_ℓ . While the full details of the experiments are presented in Section 5, we summarize some of the main findings here.

We conducted experiments on MIPLIP 2010 [KAA⁺11] instances and on randomly generated two-stage stochastic models. In the first testbed there was a small but consistent improvement in both running-time and number of iterations. More importantly, the success rate of the heuristic improved consistently. In the second testbed, the new algorithm performs even better, according to all measures. It is somewhat surprising that our small modification of the randomization step could provide noticeable improvements over the code in [FS09], specially considering that it already includes several improvements over the original Feasibility Pump (e.g. constraint propagation). In addition, the proposed modification is generic and could be easily incorporated in essentially any Feasibility Pump code. Moreover, for virtually all the seeds and instances tested the modified algorithm performed better than the original version in [FS09]; this indicates that, in practice, the modified randomization step dominates the previous one.

The rest of the paper is organized as follows: Section 3 we discuss and present our analysis of the proposed randomization scheme RANDWALKSAT_ℓ , Section 4 presents the analysis of the new randomization scheme RANDWALKSAT_ℓ in conjunction with feasibility pump, and Section 5 describes details of our empirical experiments.

Notation. We use \mathbb{R}_+ to denote the non-negative reals, and $[k] := \{1, 2, \dots, k\}$. For a vector $v \in \mathbb{R}^n$, we use $\text{supp}(v) \subseteq [n]$ to denote its support, namely the set of coordinates i where $v_i \neq 0$. We also use $\|v\|_0 = |\text{supp}(v)|$, and $\|v\|_1 = \sum_i |v_i|$ to denote the ℓ_1 norm.

3 New randomization step RandWalkSAT_ℓ

3.1 Description of the randomization step

We start by describing the WALKSAT algorithm [Sch99], that serves as the inspiration for the proposed randomization step RANDWALKSAT_ℓ , in the context of pure-binary linear programs. The vanilla version of WALKSAT starts with a random point $\bar{x} \in \{0, 1\}^n$; if this point is feasible, the algorithm returns it, and otherwise selects any constraint violated by it. The algorithm then select a random index i from the support of the selected constraint and flips the value of the entry \bar{x}_i of the solution. This process is repeated until a feasible solution is obtained. It is known that this simple algorithm finds a feasible solution in expected time at most 2^n (see [MU05] for a proof for 3-SAT instances), and Schöning [Sch99] showed that if the algorithm is restarted at every $3n$ iterations, a feasible solution is found in expected time at most a polynomial factor from $(2(1 - \frac{1}{s}))^n$, where s is the largest support size of the constraints.

Based on this WALKSAT algorithm, to obtain a randomization step for mixed-binary problems we are going to work on the projection onto the binary variables, so instead of looking for violated constraints we look for a *certificate of infeasibility* in the space of binary variables. Importantly, we use a **minimal** certificate, which makes sure that for decomposable instances the certificate does not “mix” the different blocks of the problem.

Now we proceed with a formal description of the proposed randomization step RANDWALKSAT_ℓ . Consider a mixed-binary set

$$P^I = P \cap (\{0, 1\}^n \times \mathbb{R}^d), \text{ where } P = \{(x, y) \in [0, 1]^n \times \mathbb{R}^d : Ax + By \leq b\}. \quad (3)$$

We use $\text{proj}_{\text{bin}} P$ to denote the projection of P onto the binary variables x .

Definition 3.1 (Projected certificates). *Given a mixed-binary set P^I as in (3) and a point $(\bar{x}, \bar{y}) \in \{0, 1\}^n \times \mathbb{R}^d$ such that $\bar{x} \notin \text{proj}_{\text{bin}} P$, a projected certificate for \bar{x} is an inequality $\lambda Ax + \lambda By \leq \lambda b$ with $\lambda \in \mathbb{R}_+^m$ such that: (i) \bar{x} does not satisfy this inequality; (ii) $\lambda B = 0$. A minimal projected certificate is one where the support of the vector λ is minimal (i.e. the certificate uses a minimal set of the original inequalities).*

Standard Fourier-Motzkin theory guarantees us that projected certificates always exist, and furthermore Caratheodory's theorem [Sch86] guarantees that minimal projected certificates use at most $d + 1$ inequalities. Together these give the following lemma.

Lemma 3.2. *Consider a mixed-binary set P^I as in (3) and a point $(\bar{x}, \bar{y}) \in \{0, 1\}^n \times \mathbb{R}^d$ such that $\bar{x} \notin \text{proj}_{\text{bin}} P$. There exists a vector $\lambda \in \mathbb{R}_+^m$ with support of size at most $d + 1$ such that $\lambda Ax + \lambda By \leq \lambda b$ is a minimal projected certificate for \bar{x} . Moreover, this minimal projected certificate can be obtained in polynomial-time (by solving a suitable LP).*

For completeness, see Appendix A for a proof of Lemma 3.2.

Now we can formally define the randomization step RANDWALKSAT_ℓ (notice that the condition $\lambda B = 0$ guarantees that a projected certificate has the form $ax \leq b$).

Algorithm 2 $\text{RANDWALKSAT}_\ell(\bar{x})$

- 1: // Assumes that \bar{x} does not belong to $\text{proj}_{\text{bin}} P$
 - 2: Let $ax \leq b$ be a minimal projected certificate for \bar{x}
 - 3: Sample ℓ indices from the support $\text{supp}(a)$ uniformly and independently, let \mathbf{I} be the set of indices obtained
 - 4: (Flip coordinates) For all $i \in \mathbf{I}$, set $\bar{x}_i \leftarrow 1 - \bar{x}_i$
-

Note that in the pure-binary case and $\ell = 1$, this reduces to the main step executed during WALKSAT . We remark that the flexibility of introducing the parameter ℓ will be needed in Section 4.

3.2 Analyzing the behavior of RandWalkSAT_ℓ

In this section we consider the behavior of the algorithm MBWALKSAT that tries to find a feasible mixed-binary solution by just repeatedly applying the randomization step RANDWALKSAT_ℓ .

Algorithm 3 MBWALKSAT

- 1: **input parameter:** Integer $\ell \geq 1$
 - 2: (Starting solution) Consider any mixed-binary point $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \{0, 1\}^n \times \mathbb{R}^d$
 - 3: **loop**
 - 4: **if** $\bar{\mathbf{x}}$ does not belong to $\text{proj}_{\text{bin}} P$ **then**
 - 5: $\text{RANDWALKSAT}_\ell(\bar{\mathbf{x}})$
 - 6: **else**
 - 7: (Output feasible lift of $\bar{\mathbf{x}}$) Find $\bar{\mathbf{y}} \in \mathbb{R}^d$ such that $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in P$, return $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$
 - 8: **end if**
 - 9: **end loop**
-

As mentioned in the introduction, we show that this algorithm finds a feasible solution if such exists, and the running-time improves with the sparsity and decomposability of the instance. Recall the definition of a decomposable mixed-binary problem from equation (1), and let certSupp_i denote the maximum support size of a minimal projected certificate for the instance P_i^I which consists only of the i th block.

Theorem 3.3 (Theorem 2.1 restated). *Consider a feasible decomposable mixed-binary set as in equation (1). Then with probability at least $1 - \delta$, MBWALKSAT with parameter $\ell = 1$ returns a feasible solution within $T = \lceil \ln(k/\delta) \rceil \sum_i n_i 2^{n_i \log \text{certSupp}_i}$ iterations.*

In light of Lemma 3.2, if each constraint in P_i has at most s_i integer variables, we have $\text{certSupp}_i \leq \min\{s_i \cdot (d_i + 1), n_i\}$, and thus this statement indeed implies Theorem 2.1 stated in the introduction. We remark that similar guarantees can be obtained for general ℓ , but we focus on the case $\ell = 1$ to simplify the exposition.

The high-level idea of the proof of Theorem 3.3 is the following:

1. First we show that if we run MBWALKSAT over a single block P_i^I , then with high probability the algorithm returns a feasible solution within $n_i 2^{n_i \log \text{certSupp}_i} \cdot \ln(1/\delta)$ iterations. This analysis

is inspired by the one given by Schöning [Sch99] and argues that with a small, but non-zero, probability the iteration of the algorithm makes the iterate \bar{x} closer (in Hamming distance) to a fixed solution x^* for the instance.

2. Next, we show that when running MBWALKSAT over the whole decomposable instance each iteration only depends on **one** of the blocks P_i^I ; this uses the minimality of the certificates. So in effect the execution of MBWALKSAT can be split up into independent executions over each block, and thus we can put together the analysis from Item 1 for all blocks with a union bound to obtain the result.

For the remainder of the section we prove Theorem 3.3. We start by considering a general mixed-binary set as in equation (3). Given such mixed-binary set P^I , we use $\text{certSupp} = \text{certSupp}(P^I)$ to denote the maximum support size of all minimal projected certificates.

Theorem 3.4. *Consider the execution of MBWALKSAT over a feasible mixed-binary program as in equation (3). The probability that MBWALKSAT does not find a feasible solution within the first T iterations is at most $(1-p)^{\lfloor T/n \rfloor}$, where $p = \text{certSupp}^{-n}$. In particular, for $T = n \cdot 2^{n \log(\text{certSupp})} \cdot \lceil \ln(1/\delta) \rceil$ this probability is at most δ (this follows from the inequality $(1-x) \leq e^{-x}$ valid for $x \geq 0$).*

Proof. Consider a fixed solution $x^* \in \text{proj}_{\text{bin}} P$. To analyze MBWALKSAT, we only keep track of the Hamming distance of the (random) iterate \bar{x} to x^* ; let \mathbf{X}_t denote this (random) distance at iteration t , for $t \geq 1$. If at some point this distance vanishes, i.e. $\mathbf{X}_t = 0$, we know that $\bar{x} = x^*$ and thus $\bar{x} \in \text{proj}_{\text{bin}} P$; at this point the algorithm returns a feasible solution for P^I .

Fix an iteration t . To understand the probability that $\mathbf{X}_t = 0$, suppose that in this iteration \bar{x} does not belong to $\text{proj}_{\text{bin}} P$, and let $ax \leq b$ be the minimal projected certificate for it used in RANDWALKSAT_1 . Since the feasible point x^* satisfies the inequality $ax \leq b$ but \bar{x} does not, there must be at least one index \mathbf{i}^* in the support of a such where x^* and \bar{x} differ. Then if algorithm MBWALKSAT makes a “lucky move” and chooses $\mathbf{I} = \{\mathbf{i}^*\}$ in Line 3, the modified solution after flipping this coordinate (the next line of the algorithm) is one unit closer to x^* in Hamming distance, hence $\mathbf{X}_{t+1} = \mathbf{X}_t - 1$. Moreover, since \mathbf{I} is independent of \mathbf{i} , the probability of choosing $\mathbf{I} = \{\mathbf{i}^*\}$ is $1/|\text{supp}(a)| \geq 1/\text{certSupp}$.

Therefore, if we start at iteration t and for all the next \mathbf{X}_t iterations either the iterate belongs to $\text{proj}_{\text{bin}} P$ or the algorithm makes a “lucky move”, it terminates by time $t + \mathbf{X}_t$. Thus, with probability at least $(1/\text{certSupp})^{\mathbf{X}_t} \geq (1/\text{certSupp})^n = p$ the algorithm terminates by time $t + \mathbf{X}_t \leq t + n$.

To conclude the proof, let $\alpha = \lfloor T/n \rfloor$ and call iterations $i \cdot n, \dots, (i+1) \cdot n - 1$ the i -th block of iterations. If the algorithm has not terminated by iteration $i \cdot n - 1$, then with probability at least p it terminates within the next n iterations, and hence within the i -th block. Putting these bounds together for all α blocks, the probability that the algorithm *does not* stop by the end of block α is at most $(1-p)^\alpha$. This concludes the proof. \square

Going back to decomposable problems, we now make formal the claim that minimal projected certificates for decomposable mixed-binary sets do not mix the constraints from different blocks. Notice that projected certificates for a decomposable mixed-binary set as in equation (1) have the form $\sum_i \lambda^i A^i x^i \leq \sum_i \lambda^i b^i$ and $\lambda^i B^i = 0$ for all $i \in [k]$.

Lemma 3.5. *Consider a decomposable mixed-integer set as in equation (1). Consider a point $\bar{x} \notin \text{proj}_{\text{bin}} P$ and let $\sum_i \lambda^i A^i x^i \leq \sum_i \lambda^i b^i$ be a minimal projected certificate for \bar{x} . Then this certificate uses only inequalities from one block P^j , i.e. there is j such that $\lambda^i = 0$ for all $i \neq j$. Moreover, $\bar{x}^j \notin \text{proj}_{\text{bin}} P_j$.*

Proof. Let $\bar{x} = (\bar{x}^1, \bar{x}^2, \dots, \bar{x}^k)$ and call the certificate $(ax \leq b) \triangleq (\sum_i \lambda^i A^i x^i \leq \sum_i \lambda^i b^i)$. By definition of projected certificate we have $\sum_i \lambda^i A^i \bar{x}^i > \sum_i \lambda^i b^i$, and thus by linearity there must be an index j such that $\lambda^j A^j \bar{x}^j > \lambda^j b^j$. Moreover, as remarked earlier, decomposability implies that the certificate satisfies $\lambda^i B^i = 0$ for all i , so in particular for j . Thus, the inequality $\lambda^j (A^j, B^j)(x^j, y^j) \leq \lambda^j b^j$ obtained by combining only the inequalities from P_j is a projected certificate for \bar{x} . The minimality of the original certificate $ax \leq b$ implies that $\lambda^i = 0$ for all $i \neq j$. This concludes the first part of the proof.

Moreover, since $\lambda^j A^j \bar{x}^j > \lambda^j b^j$ and $\lambda^j B^j = 0$ we have that $\lambda^j (A^j, B^j)(\bar{x}^j, y) > \lambda^j b^j$ for all y , and hence \bar{x}^j does not belong to $\text{proj}_{\text{bin}} P_j$. This concludes the proof. \square

We can finally prove the desired theorem.

Proof of Theorem 3.3. We use the natural decomposition $\bar{\mathbf{x}} = (\bar{\mathbf{x}}^1, \dots, \bar{\mathbf{x}}^k) \in \{0, 1\}^{n_1} \times \dots \times \{0, 1\}^{n_k}$ of the iterates of the algorithm. From Lemma 3.5, we have that for each scenario, each iteration of MBWALKSAT is associated with just one of the blocks P_j^I 's, namely the P_j^I containing all the inequalities in the minimal projected certificate used in this iteration; let $\mathbf{J}_t \in [k]$ denote the (random) index j of the block associated to iteration t . Notice that at iteration t , only the binary variables $x^{\mathbf{J}_t}$ can be modified by the algorithm.

Let $T_i = n_i 2^{n_i \log n_i} \lceil \ln(k/\delta) \rceil$. Applying the proof of Theorem 3.4 to the iterations $\{t : \mathbf{J}_t = i\}$ with index i , we get that with probability at least $1 - \frac{\delta}{k}$ the algorithm finds some $\bar{\mathbf{x}}^i$ in $\text{proj}_{\text{bin}} P_i$ within the first T_i of these iterations. Moreover, after the algorithm finds such a point, it does not change it (that is, the remaining iterations have index $\mathbf{J}_t \neq i$, due to the second part of Lemma 3.5).

Therefore, by taking a union bound we get that with probability at least $1 - \delta$, for all $i \in [k]$ the algorithm finds $\bar{\mathbf{x}}^i \in \text{proj}_{\text{bin}} P_i$ within the first T_i iterations with index i (for a total of $\sum_i T_i = T$ iterations). When this happens, the total solution $\bar{\mathbf{x}}$ belongs to $\text{proj}_{\text{bin}} P$ and the algorithm returns. This concludes the proof. \square

4 Randomization step RandWalkSAT $_\ell$ within Feasibility Pump

In this section we incorporate the randomization step RANDWALKSAT $_\ell$ into the Naïve Feasibility Pump, the resulting algorithm being called WFP. We describe this algorithm in a slightly different way and using a notation more convenient for the analysis.

Consider a mixed-binary set P^I as in equation (3). Given a 0/1 point $\tilde{x} \in \{0, 1\}^n$, let $\ell_1\text{-proj}(P, \tilde{x})$ denote a point (x, y) in P where $\|\tilde{x} - x\|_1$ is as small as possible. Also, for a vector $v \in [0, 1]^p$, we use $\text{round}(v)$ to denote the vector obtained by rounding each component of v to the closest integer; we use the convention that $\frac{1}{2}$ is rounded to 1, but any consistent rounding would suffice. Notice that operations ‘ $\ell_1\text{-proj}$ ’ and ‘round’ correspond precisely to Steps 5 and 4 in the Naïve Feasibility Pump. With this notation, algorithm WFP can be described as follows.

Algorithm 4 WFP

```

1: input parameter: integer  $\ell \geq 1$ 
2: Let  $(\bar{x}^0, \bar{y}^0)$  be an optimal solution of the LP relaxation
3: Let  $\tilde{x}^0 = \text{round}(\bar{x}^0)$ 
4: for  $t = 1, 2, \dots$  do
5:    $(\bar{\mathbf{x}}^t, \bar{\mathbf{y}}^t) = \ell_1\text{-proj}(P, \tilde{\mathbf{x}}^{t-1})$ 
6:    $\tilde{\mathbf{x}}^t = \text{round}(\bar{\mathbf{x}}^t)$ 
7:   if  $(\tilde{\mathbf{x}}^t, \bar{\mathbf{y}}^t) \in P$  then ▷ equivalently,  $\tilde{\mathbf{x}}^t \in \text{proj}_{\text{bin}}(P)$ 
8:     Return  $(\tilde{\mathbf{x}}^t, \bar{\mathbf{y}}^t)$ 
9:   end if
10:  if  $\tilde{\mathbf{x}}^t = \tilde{\mathbf{x}}^{t-1}$  then ▷ iterations have stalled
11:     $\tilde{\mathbf{x}}^t = \text{RANDWALKSAT}_\ell(\tilde{\mathbf{x}}^t)$ 
12:  end if
13: end for

```

Note that stalling in the above algorithm is determined using the condition $\tilde{\mathbf{x}}^t = \tilde{\mathbf{x}}^{t-1}$. What about ‘long cycle’ stalling, that is $\tilde{\mathbf{x}}^t = \tilde{\mathbf{x}}^{t'}$ where $t' < t - 1$, but $\tilde{\mathbf{x}}^{t'}, \dots, \tilde{\mathbf{x}}^{t-1}$ are all distinct binary vectors. As it turns out (assuming no numerical errors) a consistent rounding rule implies that stalling will always occur with cycles of length two.

Theorem 4.1. *With consistent rounding, long cycles cannot occur.*

We present a proof of 4.1 in Appendix C. For the remainder of the section, we analyze the behavior of algorithm WFP on separable subset-sum instances, proving Theorem 2.4 stated in the introduction.

4.1 Running time of WFP for separable subset-sum instances: Proof of Theorem 2.4

Notice that the projection operators ‘ ℓ_1 -proj’ and ‘round’ now present also act on each block independently, namely given a point $x = (x^1, \dots, x^k) \in \mathbb{R}^{n_1} \times \dots \times \mathbb{R}^{n_k}$, if $(\tilde{x}^1, \dots, \tilde{x}^k) = \ell_1\text{-proj}(P, x)$ then $\tilde{x}^i = \ell_1\text{-proj}(P_i, x^i)$ for all $i \in [k]$, and similarly for ‘round’. Therefore, as in the proof of Theorem 3.3, it suffices to analyze the execution of algorithm WFP over a single block/inequality of the separable subset-sum problem. More precisely, it suffices to prove the following guarantee for WFP on a general subset-sum instance.

Theorem 4.2. *Consider a feasible subset-sum problem $P \subseteq \mathbb{R}^n$. Then for every $T \geq 1$, the probability that WFP with $\ell = 2$ does not find a feasible solution within the first $2T$ iterations is at most $(1-p)^{\lfloor T/n \rfloor}$, where $p = (1/n^2)^n$. In particular, for $T = n \cdot 2^{2n \log n} \cdot \lceil \ln(1/\delta) \rceil$ this probability is at most δ .*

The high-level idea of the proof of this theorem is the following. We use a similar strategy as before, where we consider a fixed feasible solution x^* and track its distance to the iterates \tilde{x}^t generated by algorithm WFP. However, while again the randomization step RANDWALKSAT_2 brings \tilde{x}^t closer to x^* with small but non-zero probability, the issue is that the projections ‘ ℓ_1 -proj’ and ‘round’ in the next iterations could send the iterate even further from x^* . To analyze the algorithm we then use the structure of subset-sum instances to: 1) First control the combination ‘ ℓ_1 -proj + round’ in Steps 5 and 6, showing that in this case they are *idempotent*, namely applying them once or repeatedly yields the same effect (Lemma 4.3); 2) Strengthen the analysis of Theorem 3.3 to show that a round of RANDWALKSAT_2 *plus* ‘ ℓ_1 -proj + round’ still has a non-zero probability of generating a point closer to x^* (Lemma 4.6). For this, it will be actually important that we use $\ell = 2$ in algorithm WFP (actually $\ell \geq 2$ suffices).

For the remainder of the section we prove Theorem 4.2. To simplify the notation we omit the polytope P from the notation of ℓ_1 -proj. We assume that our subset-sum problem $P = \{x \in [0, 1]^n : ax = b\}$ is such that *all* coordinates of a are positive, since components with $a_i = 0$ do not affect the problem (more precisely, after the first iteration of the algorithm, the value of \tilde{x}_i^t is set to 0 or 1 and does not change anymore, and this value does not affect the feasibility of the solutions \tilde{x}^t 's). Also remember that subset-sum problems only have binary variables.

Given a point $\tilde{x} \in \{0, 1\}^n$, let $\text{AltProj}(\tilde{x}) \in \{0, 1\}^n$ be the effect of applying to \tilde{x} ℓ_1 -proj(.) and then round(.). Notice that if \tilde{x} belongs to P , then $\text{AltProj}(\tilde{x}) = \tilde{x}$. Then algorithm WFP can be thought as performing a AltProj operation, then checking if the iterate obtained either belongs to P (in which case it exits) or if it equals the previous iterate (in which case it applies RANDWALKSAT_2); if neither of these occur, then another AltProj operation is performed. So an important component for analyzing this algorithm is getting a good control over a sequence of AltProj operations. For that, define the iterated operation $\text{AltProj}^t(\tilde{x}) = \text{AltProj}(\text{AltProj}^{t-1}(\tilde{x}))$ (with $\text{AltProj}^1 = \text{AltProj}$) and if the sequence $(\text{AltProj}^t(\tilde{x}))$ stabilizes at a point, let $\text{AltProj}^*(\tilde{x})$ denote this point.

A crucial observation, given by the next lemma, is that for subset-sum instances the operation of AltProj is idempotent, namely it stabilizes after just one operation.

Lemma 4.3. *Let P be a subset-sum instance. Then for every $\tilde{x} \in \{0, 1\}^n$, $\text{AltProj}_P^*(\tilde{x}) = \text{AltProj}_P(\tilde{x})$.*

Proof. Again to simplify the notation we omit the polyhedron P when writing ℓ_1 -proj and AltProj . Let $\bar{x} = \ell_1\text{-proj}(\tilde{x})$ and recall it is an extreme point of P . Clearly, if $\tilde{x} \in P$ then $\text{AltProj}(\tilde{x}) = \tilde{x}$ and hence $\text{AltProj}^*(\tilde{x}) = \text{AltProj}(\tilde{x})$. Similarly, if \bar{x} is a 0/1 point then $\text{AltProj}(\tilde{x}) = \bar{x}$, and again $\text{AltProj}^*(\tilde{x}) = \text{AltProj}(\tilde{x})$.

Thus, assume that $\tilde{x} \notin P$ and \bar{x} is not a 0/1 point. Since \bar{x} is an extreme point of the subset-sum LP P it has exactly 1 fractional coordinate, so by permuting indices we assume without loss of generality:

1. $\bar{x}_1 = \dots = \bar{x}_k = 1$.
2. $\bar{x}_{k+1} \in (0, 1)$.
3. $\bar{x}_{k+2} = \dots = \bar{x}_n = 0$
4. $a_{k+2} \geq a_{k+3} \geq \dots \geq a_n$.
5. $a_1 \leq a_2 \leq a_3 \leq \dots \leq a_k$.

Now we look at the points obtained after applying $\text{round}(\cdot)$ and $\ell_1\text{-proj}(\cdot)$ to \bar{x} , namely let $\tilde{x}' := \text{round}(\bar{x}) = \text{AltProj}(\tilde{x})$ and let $\bar{x}' := \ell_1\text{-proj}(\tilde{x}')$. Notice that \bar{x}' is obtained by solving:

$$\begin{aligned} \min \quad & \sum_{\{j \mid \tilde{x}'_j=0\}} x_j + \sum_{\{j \mid \tilde{x}'_j=1\}} (1 - x_j) \\ \text{s.t.} \quad & ax = b \\ & 0 \leq x \leq 1. \end{aligned} \tag{4}$$

Case 1: $\bar{x}_{k+1} < 1/2$. Then $\tilde{x}'_i = 1$ for all $i \leq k$, $\tilde{x}'_i = 0$ for all $i \geq k+1$; also notice $\tilde{x}' \leq \bar{x}$, and hence $a\tilde{x}' < b$; thus \bar{x}' is obtained from \tilde{x}' by increasing some components of 0 value. We have three subcases:

- a. If $a_{k+1} > a_{k+2}$: then a_{k+1} is the largest coordinate of a where \tilde{x}' has value 0, so it follows from (4) that \bar{x}' is obtained from \tilde{x}' by raising its $(k+1)$ -component from 0 to \bar{x}_{k+1} . Thus, $\bar{x}' = \bar{x}$, and hence $\text{AltProj}(\text{AltProj}(\tilde{x})) = \text{round}(\bar{x}') = \text{round}(\bar{x}) = \text{AltProj}(\tilde{x})$; this implies $\text{AltProj}^*(\tilde{x}) = \text{AltProj}(\tilde{x})$.
- b. If $a_{k+1} < a_{k+2}$: then \bar{x}' is obtained from \tilde{x} by raising its $(k+2)$ -component to a value that is at most $\bar{x}_{k+1} < 1/2$. Now, $\text{round}(\bar{x}') = \tilde{x}'$, so again we get $\text{AltProj}(\text{AltProj}(\tilde{x})) = \text{round}(\bar{x}') = \tilde{x}' = \text{AltProj}(\tilde{x})$ and we are done.
- c. If $a_{k+1} = a_{k+2}$: Since \bar{x}' is a vertex of the subset-sum LP P , again it only has 1 fractional component (either $k+1$ or $k+2$) and then it is easy to see that \bar{x}' is equal to the one in either Case (a) or Case (b) above; thus the result also holds for this case.

Case 2: $\bar{x}_{k+1} \geq 1/2$. Then \tilde{x}' is such that $\tilde{x}'_i = 1$ for all $i \leq k+1$ and $\tilde{x}' = 0$ for all $i \geq k+2$; also notice $\tilde{x}' \geq \bar{x}$ and hence $a\tilde{x}' > b$. Now, consider $\bar{x}' = \ell_1\text{-proj}(\tilde{x}')$:

- a. If $a_k < a_{k+1}$: This is analogous to Case 1a: \bar{x}' is obtained by lowering the $(k+1)$ -coordinate of \tilde{x}' from 1 to \bar{x} , and thus $\bar{x}' = \bar{x}$; the rest of the proof is identical to Case 1a.
- b. If $a_k > a_{k+1}$: In this case, \bar{x}' is obtained by lowering the k -component of \tilde{x}' . Since $a\bar{x} = a\bar{x}' = b$, and k and $(k+1)$ are the only components where \bar{x} and \bar{x}' differ, we have: $a_k + a_{k+1}\bar{x}_{k+1} = a_k\bar{x}'_k + a_{k+1}$. Hence $\bar{x}'_k = 1 - \frac{a_{k+1}}{a_k}(1 - \bar{x}_{k+1}) \geq 1/2$ and $\text{round}(\bar{x}') = \tilde{x}'$; the rest of the proof is identical to Case 1b.
- c. If $a_k = a_{k+1}$: Identical to Case 1c.

□

Therefore, there is not much loss in looking at a “compressed” version of algorithm WFP that packs repeated applications of AltProj until stalling happens into a single AltProj^* ; more formally, we have the following algorithm (stated in the pure-binary case to simplify the notation).

Algorithm 5 WFP-Compressed

```

1: input parameter: integer  $\ell \geq 1$ 
2: Let  $\bar{x}^0$  be an optimal solution of the LP relaxation
3: Let  $\tilde{z}^0 = \text{round}(\bar{x}^0)$ 
4: for  $\tau = 1, 2, \dots$  do
5:    $\tilde{z}^\tau = \text{AltProj}^*(\tilde{z}^{\tau-1})$ 
6:   if  $\tilde{z}^\tau \in P$  then
7:     Return  $\tilde{z}^\tau$ 
8:   end if
9:    $\tilde{z}^\tau = \text{RANDWALKSAT}_\ell(\tilde{z}^\tau)$ 
10: end for
```

Intuitively, Lemma 4.3 should imply that packing the repeated applications of AltProj into a single AltProj^* should not save more than 1 iteration. To see this more formally, assume that both algorithms

use as starting point the same optimal solution of the LP, so $\tilde{z}^0 = \tilde{x}^0$. Now condition on a scenario where we have $\tilde{z}^\tau = \tilde{x}^t$ at the beginning of iterations τ and t of algorithms WFP-Compressed and WFP respectively (for $\tau, t \geq 1$). Then we claim that either both algorithms return at the current iteration, or $\tilde{z}^{\tau+1}$ has the same distribution as either \tilde{x}^{t+1} or \tilde{x}^{t+2} (at the beginning of they respective iterations): If $\tilde{z}^\tau = \tilde{x}^t \in P$, then both algorithms return; if $\tilde{x}^t \notin P$ but $\tilde{x}^t = \tilde{x}^{t-1}$, then both algorithms WFP-Compressed and WFP employ RANDWALKSAT_2 over $\tilde{z}^\tau = \tilde{x}^t$, in which case $\tilde{z}^{\tau+1}$ has the same distribution as \tilde{x}^{t+1} ; finally, if $\tilde{x}^t \neq \tilde{x}^{t-1}$, then WFP at the beginning of the next iteration will have $\tilde{x}^{t+1} = \text{AltProj}(\tilde{x}^t)$, which by Lemma 4.3 (and $t \geq 1$) equals \tilde{x}^t itself, and so it will employ RANDWALKSAT_2 to $\tilde{x}^{t+1} = \tilde{x}^t$ and again we have that \tilde{x}^{t+2} has the same distribution as $\tilde{z}^{\tau+1}$.

Therefore, since we can employ this argument to couple iterations $\leq \tau$ of WFP-Compressed with iterations $\leq 2\tau$ of WFP, we have the following result.

Lemma 4.4. *Consider the application of algorithms WFP and WFP-Compressed over the subset-sum problem P . Then the probability that algorithm WFP returns after at most $2T$ iterations is at least the probability that algorithm WFP-Compressed after at most T iterations.*

Therefore, it suffices to upper bound the number of iterations of WFP-Compressed until it returns. To avoid ambiguity, let z^τ be the value of \tilde{z}^τ at the *beginning* of iteration τ of WFP-Compressed. Notice that $z^1 = \text{AltProj}^*(\tilde{x}^0)$, and $z^{\tau+1} = \text{AltProj}^*(\text{RANDWALKSAT}_2(z^\tau))$ for $\tau \geq 2$. It suffices to show that with probability at least $1 - (1-p)^{T/n}$, there is $\tau \leq T/2$ such that z^τ belongs to P .

To do so, for $\tilde{x} \in \{0, 1\}^n$ and $I \subseteq [n]$ let $\text{flip}(\tilde{x}, I)$ denote the 0/1 vector obtained starting from \tilde{x} and flipping the value of all coordinates that belongs to I . Notice that (up to scaling) the only possible projected certificates for our subset-sum problem are $ax \geq b$ and $ax \leq b$. Since we have assumed that the vector a has full support, it follows that on this problem $\text{RANDWALKSAT}_2(\tilde{x}) = \text{flip}(\tilde{x}, \mathbf{I})$ for \mathbf{I} being the set obtained by sampling independently two indices uniformly from $[n]$.

The next lemma then shows that there is always a “lucky choice” of set \mathbf{I} in $\text{RANDWALKSAT}_2(z^\tau)$ that brings $z^{\tau+1} = \text{AltProj}^*(\text{RANDWALKSAT}_2(z^\tau))$ closer to a fixed solution x^* to the subset-sum problem.

The following definition is convenient.

Definition 4.5. *A point $\tilde{x} \in \{0, 1\}^n$ is called a stalling solution if $\text{AltProj}(\tilde{x}) = \tilde{x}$.*

Lemma 4.6. *Let $x^* \in \{0, 1\}^n$ be a feasible solution to the subset-sum problem. Consider $\tilde{x} \in \{0, 1\}^n$ with $a\tilde{x} \neq b$ that satisfies the fixed point condition $\text{AltProj}(\tilde{x}) = \tilde{x}$. Then there is a set $I \subseteq [n]$ of size at most 2 such that the point $x' = \text{AltProj}_P^*(\text{flip}(\tilde{x}, I))$ is closer to x^* than \tilde{x} , namely $\|x' - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 1$.*

Proof. Again to simplify the notation we omit P from $\ell_1\text{-proj}$ and AltProj , and use $\text{flip}(\tilde{x}, j)$ instead of $\text{flip}(\tilde{x}, \{j\})$ in the singleton case.

We start with a couple of claims.

Claim 1 Suppose $\tilde{x} \in \{0, 1\}^n$ is a stalling point. If $a\tilde{x} < b$, then there is $k \notin \text{supp}(\tilde{x})$ such that $\ell_1\text{-proj}(\tilde{x})_i = \tilde{x}_i$ for all $i \neq k$, and $\ell_1\text{-proj}(\tilde{x})_k \in (0, \frac{1}{2})$. Similarly, if $a\tilde{x} > b$, then there is $k \in \text{supp}(\tilde{x})$ such that $\ell_1\text{-proj}(\tilde{x})_i = \tilde{x}_i$ for all $i \neq k$, and $\ell_1\text{-proj}(\tilde{x})_k \in (\frac{1}{2}, 1)$.

Proof of Claim 1. We only prove the first statement, the proof of the second is completely analogous. Since \tilde{x} is stalling we have that $\text{round}(\ell_1\text{-proj}(\tilde{x})) = \tilde{x}$, and since $\ell_1\text{-proj}(\tilde{x})$ is an extreme point of the subset-sum problem P it has at most 1 fractional component, and hence only differs in one component k from

$$\text{round}(\ell_1\text{-proj}(\tilde{x})) = \tilde{x}.$$

Since $a \cdot \ell_1\text{-proj}(\tilde{x}) = b > a \cdot \tilde{x}$, we have that $\tilde{x}_k = 0$ and $\ell_1\text{-proj}(\tilde{x})_k > 0$; since $\text{round}(\ell_1\text{-proj}(\tilde{x})_k) = \tilde{x}_k = 0$, we have $\ell_1\text{-proj}(\tilde{x})_k < \frac{1}{2}$. \square

Claim 2 Consider a point $\tilde{x} \in \{0, 1\}^n$.

1. If the objective value of (4) is strictly less than $\frac{1}{2}$, then $\text{AltProj}(\tilde{x}) = \tilde{x}$.
2. If the objective value of (4) is strictly less than 1, then $\|\text{AltProj}(\tilde{x}) - \tilde{x}\|_0 \leq 1$.

Proof of Claim 2. Let $\bar{x} = \ell_1\text{-proj}(\tilde{x})$ be an optimal solution for (4). Proof of Part 1: the assumption implies that $|\bar{x}_i - \tilde{x}_i| < \frac{1}{2}$ for all i , which directly implies that $\text{AltProj}(\tilde{x}) = \text{round}(\bar{x}) = \tilde{x}$.

Proof of Part 2: the assumption implies that there can be at most one index j with $|\bar{x}_j - \tilde{x}_j| \geq \frac{1}{2}$, which implies that for all $i \neq j$, $\text{AltProj}(\tilde{x})_i = \text{round}(\bar{x}_i) = \tilde{x}_i$ and the result follows. \square

Now we are ready to present the proof of Lemma 4.6. Let x^* and \tilde{x} be as in the statement of the Lemma. From Lemma 4.3 we know that

$$\text{AltProj}^*(\text{flip}(\tilde{x}, J)) = \text{AltProj}(\text{flip}(\tilde{x}, J)),$$

so it suffices to work with the right-hand side instead. Since $\tilde{x} \neq x^*$ we have $\text{supp}(\tilde{x}) \neq \text{supp}(x^*)$. We separate the proof in three cases depending on the relationship between these supports.

Case 1: $\text{supp}(\tilde{x}) \subsetneq \text{supp}(x^*)$: Pick any $j \in \text{supp}(x^*) \setminus \text{supp}(\tilde{x})$ and notice that $\|\text{flip}(\tilde{x}, j) - x^*\|_0 = \|\tilde{x} - x^*\|_0 - 1$. Notice that both $\text{supp}(\tilde{x})$ and $\text{supp}(\text{flip}(\tilde{x}, j))$ are contained in the support of x^* , and hence we have $a\tilde{x} \leq b$ and $a \cdot \text{flip}(\tilde{x}, j) \leq b$. Moreover, since $\text{flip}(\tilde{x}, j) \geq \tilde{x}$, it is easy to see that the optimal value of (4) for $\text{flip}(\tilde{x}, j)$ is *strictly less than* that for \tilde{x} (we need to raise fewer variables to make the point satisfy $ax = b$), which by Claim 1 is at most $\frac{1}{2}$. Thus, employing Part 1 of Claim 2 to $\text{flip}(\tilde{x}, j)$ gives that $\text{AltProj}(\text{flip}(\tilde{x}, j)) = \text{flip}(\tilde{x}, j)$, which is the desired point closer to x^* .

Case 2: $\text{supp}(x^*) \subsetneq \text{supp}(\tilde{x})$: The proof is the same as above, with the only change that we take $j \in \text{supp}(\tilde{x}) \setminus \text{supp}(x^*)$.

Case 3: The supports $\text{supp}(x^*)$ and $\text{supp}(\tilde{x})$ are not contained in one another. In this case $a\tilde{x}$ can be either $< b$ or $> b$:

1. If $a\tilde{x} < b$. Take $m \in \text{supp}(x^*) \setminus \text{supp}(\tilde{x})$. If $a \cdot \text{flip}(\tilde{x}, m) \leq b$, then we can argue exactly as in Case 1 to get that $\text{AltProj}(\text{flip}(\tilde{x}, m)) = \text{flip}(\tilde{x}, m)$, which is closer to x^* than \tilde{x} . So consider the case $a \cdot \text{flip}(\tilde{x}, m) > b$. Take $i \in \text{supp}(\tilde{x}) \setminus \text{supp}(x^*)$ and consider $\text{flip}(\tilde{x}, \{m, i\})$, which is 2 units closer to x^* in Hamming distance.

We claim that the optimal value of (4) for $\text{flip}(\tilde{x}, \{m, i\})$ is strictly less than 1. Suppose $a \cdot \text{flip}(\tilde{x}, \{m, i\}) \leq b$; since $a \cdot \text{flip}(\tilde{x}, m) > b$ (notice $\text{flip}(\tilde{x}, m)$ is obtained from $\text{flip}(\tilde{x}, \{m, i\})$ by increasing coordinate i to 1), this means that we can make $\text{flip}(\tilde{x}, \{m, i\})$ satisfy $ax = b$ by increasing coordinate i to a value *strictly less* than 1, thus upper bounding the optimum of (4). On the other hand, consider $a \cdot \text{flip}(\tilde{x}, \{m, i\}) > b$; notice $a \cdot \text{flip}(\tilde{x}, i) \leq a \cdot \tilde{x} < b$ (the last uses a running assumption), and thus again we can make $\text{flip}(\tilde{x}, \{m, i\})$ satisfy $ax = b$ by decreasing coordinate m to a value strictly smaller than 1. This proves the claim.

With this claim in place, we can just employ Part 2 of Claim 2 to $\text{flip}(\tilde{x}, \{m, i\})$ and triangle inequality to obtain that $\|\text{AltProj}(\text{flip}(\tilde{x}, \{m, i\})) - x^*\|_0$ is at most

$$1 + \|\text{flip}(\tilde{x}, \{m, i\}) - x^*\|_0 = 1 + \|\tilde{x} - x^*\|_0 - 2,$$

which gives the desired result.

2. If $a\tilde{x} > b$. The proof of this case mirrors that of the above case (only with the inequalities $<$ and $>$ reversed throughout).

\square

Notice that since z^τ is obtained from $\text{AltProj}^*(.)$, it satisfies the fixed point condition $\text{AltProj}(z^\tau) = z^\tau$. Thus, as long as z^τ does not belong to P we can apply the above lemma to obtain that with probability at least $\frac{1}{n^2}$ we have \mathbf{I} in RANDWALKSAT_2 equal to the set I in the lemma and thus the iterate moves closer to a feasible solution; more formally we have the following.

Corollary 4.7. *Let $x^* \in \{0, 1\}^n$ be a feasible solution to the subset-sum problem P . Then*

$$\Pr\left(\|z^{\tau+1} - x^*\|_0 \leq \|z^\tau - x^*\|_0 - 1 \mid z^\tau \notin P\right) \geq \frac{1}{n^2}.$$

Now we can conclude the proof of Theorem 4.2 arguing just like in the proof of Theorem 3.4.

Proof of Theorem 4.2. Consider $x^* \in P$ and let $\mathbf{Z}_\tau = \|\mathbf{z}^\tau - x^*\|_0$. Notice that $\mathbf{Z}_\tau = 0$ implies $\mathbf{z}^\tau = x^*$ and hence $\mathbf{z}^\tau \in P$. Corollary 4.7 gives that $\Pr(\mathbf{Z}_{\tau+1} \leq \mathbf{Z}_\tau - 1 \mid \mathbf{z}^\tau \notin P) \geq \frac{1}{n^2}$. Therefore, if we start at iteration τ and for all the next \mathbf{Z}_τ iterations either the iterate $\mathbf{z}^{\tau'}$ belongs to P or the algorithm reduces $\mathbf{Z}_{\tau'}$, it terminates by time $\tau + \mathbf{Z}_\tau$. Thus, with probability at least $(1/n^2)^{\mathbf{Z}_\tau} \geq (1/n^2)^n = p$ the algorithm terminates by time $t + \mathbf{Z}_\tau \leq t + n$.

To conclude the proof, let $\alpha = \lfloor T/n \rfloor$ and call time steps $i \cdot n, \dots, (i+1) \cdot n - 1$ the i -th block of time. From the above paragraph, the probability that there is τ in the i th block of time such that $\mathbf{z}^\tau \in P$ conditioned on $\mathbf{z}^{i \cdot n - 1} \notin P$ is at least p . Using the chain rule of probability gives that the probability that there is no $\mathbf{z}^\tau \in P$ within any of the α blocks is at most $(1-p)^\alpha$. This concludes the proof. \square

5 Computations

In this section, we describe the algorithms that we have implemented and report computational experiments comparing the performance of the original Feasibility Pump 2.0 algorithm from [FS09], which we denote by FPORIG, to our modified code that uses the new perturbation procedure. The code is based on the current version of the Feasibility Pump 2.0 code (the one available on the NEOS servers), which is implemented in C++ and linked to IBM ILOG CPLEX 12.6.3 [ILO] for preprocessing and solving LPs. All features such as constraint propagation which are part of the Feasibility Pump 2.0 code have been left unchanged.

All algorithms have been run on a cluster of identical machines, each equipped with an Intel Xeon CPU E3-1220 V2 running at 3.10GHz and 16 GB of RAM. Each run had a time limit of half an hour.

5.1 WalkSAT-based perturbation

In preliminary tests, we implemented the algorithm WFP as described in the previous section. However, its performance was not competitive with FPORIG. In hindsight, this can be justified by the following reasons:

- Picking a fixed ℓ can be tricky. Too small or too big a value can lead to slow convergence in practical implementations.
- Using RANDWALKSAT $_\ell$ at each perturbation step can be overkill, as in most cases the original perturbation scheme does just fine.
- Computing the minimal certificate is too expensive, as it requires solving LPs.

For the reasons above, we devised a more conservative implementation of a perturbation procedure inspired by WALKSAT, which we denote by WFPBASE. The algorithm works as follows. Let $F \subset [n]$ be the set of indices with positive fractionality $|\tilde{x}_j - \bar{x}_j|$. If $TT \leq |F|$, then the perturbation procedure is just the original one in FPORIG. Else, let S be the union of the supports of the constraints that are not satisfied by the current point (\tilde{x}, \bar{y}) . We select the $|F|$ indices with largest fractionality $|\tilde{x}_j - \bar{x}_j|$ and select uniformly at random $\min\{|S|, TT - |F|\}$ indices from S , and flip the values in \tilde{x} for all the selected indices.

Note also that the above procedure applies only to the case in which a cycle of length one is detected. In case of longer cycle, we use the very same restart strategy of FPORIG.

5.2 Computational results

We tested the two algorithms on two classes of models: two-stage stochastic models, and the MIPLIB 2010 dataset.

Two-stage stochastic models. In order to validate the hypothesis suggested by the theoretical results that our walkSAT-based perturbation should work well on almost-decomposable models, we tested WFPBASE on two-stage stochastic models. These are the deterministic equivalent of two-stage stochastic

programs and have the form

$$\begin{aligned} Ax + D^i y^i &\leq b^i, i \in \{1, \dots, k\} \\ x &\in \{0, 1\}^p \\ y^i &\in \{0, 1\}^q, i \in \{1, \dots, k\}. \end{aligned}$$

The variables x are the first-stage variables, and y^i are the second-stage variables for the i th scenario. Notice that these second-stage variables are different for each scenario, and are only coupled through the first-stage variables x . Thus, as long as the number of scenarios is reasonably large compared to dimensions of x, y^1, \dots, y^k , these problems are to some extent almost-decomposable.

For our experiments we randomly generated instances of this form as follows: (1) the entries in A and the D^i 's are independently and uniformly sampled from $\{-10, \dots, 10\}$; (2) to guarantee feasibility, a 0/1 point is sampled uniformly at random from $\{0, 1\}^{p+k \cdot q}$ and the right-hand sides b^i are set to be the smallest ones that make this points feasible. We generated 50 instances, 5 for each setting of parameters $k = \{5, 15, 25, 35, 45\}$, $p = \{10, 20\}$, $q = 10$.

We compared the two algorithms FPORIG and WFPBASE over these instances using ten different random seeds. A seed by seed comparison is reported in Table 1. In the tables, **#found** denotes the number of models for which a feasible solution was found, while **time** and **itr.** report the shifted geometric means [Ach07] of running times and iterations, respectively.

Seed	# found		time (s)		itr.	
	FPORIG	WFPBASE	FPORIG	WFPBASE	FPORIG	WFPBASE
1	28	31	4.12	3.36	124.43	76.02
2	26	35	4.06	3.17	122.51	82.85
3	25	37	4.00	3.02	117.74	72.50
4	26	36	4.28	3.40	119.82	75.17
5	25	31	4.20	3.44	124.41	81.66
6	26	35	3.98	3.56	122.74	79.73
7	25	27	4.22	3.98	126.77	91.59
8	28	38	3.82	3.10	112.91	73.92
9	25	31	4.22	3.67	117.61	83.46
10	25	32	4.12	3.57	116.92	88.23

Table 1: Aggregated results on two-stage stochastic models.

Notice that WFPBASE performed substantially better than FPORIG, in agreement with our theoretical results. Using the walkSAT-based perturbation the average number of successful instances increased by 28%, while average runtime was reduced by 17% and average number of iterations was reduced by 33%.

MIPLIB 2010. We also compared the algorithms on a subset of models from MIPLIB 2010 [KAA⁺11]. The subset is defined by the models for which at least one of the two algorithms took more than 20 iterations to find a feasible solution (if any); the remaining models are basically too easy and not useful for comparing the two perturbation procedures. We are thus left with a subset of 82 models. Again we compared the two algorithms using ten different random seeds. A seed by seed comparison is reported in Table 2.

Even though the improvement in this heterogeneous testbed was less dramatic as in the two-stage stochastic models, as expected, WFPBASE still consistently dominates FPORIG: it can find more solutions in 7 out 10 cases (in the remaining 3 cases it is a tie), taking always less time and almost always fewer iterations. On average over the seeds, WFPBASE increased the number of successfully solved instances by 6%, reduced by the computation time by 8.4% and reduced the number of iterations by 5.9%.

In conclusion, given that the suggested modification is very simple to implement, and appears to dominate FPORIG consistently, it suggests it is a good idea to add it as a feature in all future feasibility pump codes.

Seed	# found		time (s)		itr.	
	FPORIG	WFPBASE	FPORIG	WFPBASE	FPORIG	WFPBASE
1	33	34	1070.35	1068.09	103.38	104.59
2	34	34	1073.03	1004.84	108.65	104.05
3	34	39	1125.44	976.16	107.10	96.18
4	34	36	1045.10	976.31	101.30	96.24
5	31	32	1033.60	974.56	96.67	94.36
6	34	34	974.47	880.05	99.61	91.20
7	33	36	972.96	877.45	102.39	95.04
8	29	32	1085.82	1049.22	104.63	103.22
9	37	37	1065.50	937.19	101.44	91.73
10	32	37	1096.99	913.50	103.01	90.85

Table 2: Aggregated results on MIPLIB2010.

Acknowledgments

We would like to thank Andrea Lodi for discussions and clarifications on Feasibility Pump. Santanu S. Dey and Andres Iroume would like to gratefully acknowledge the support of NSF grants CMMI 1562578 and CMMI 1149400 respectively.

References

- [AB07] Tobias Achterberg and Timo Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007.
- [Ach07] Tobias Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.
- [BCLM09] Pierre Bonami, Gérard Cornuéjols, Andrea Lodi, and François Margot. A feasibility pump for mixed integer nonlinear programs. *Math. Program.*, 119(2):331–352, 2009.
- [BEE⁺14] Natasha L. Boland, Andrew C. Eberhard, Faramroze G. Engineer, Matteo Fischetti, Martin W. P. Savelsbergh, and Angelos Tsoukalas. Boosting the feasibility pump. *Math. Program. Comput.*, 6(3):255–279, 2014.
- [BEET12] Natasha L. Boland, Andrew C. Eberhard, Faramroze G. Engineer, and Angelos Tsoukalas. A new approach to the feasibility pump in mixed integer programming. *SIAM Journal on Optimization*, 22(3):831–861, 2012.
- [BFL07] Livio Bertacco, Matteo Fischetti, and Andrea Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63–76, 2007.
- [DFLL10] Claudia D’Ambrosio, Antonio Frangioni, Leo Liberti, and Andrea Lodi. Experiments with a feasibility pump approach for nonconvex minlps. In *Experimental Algorithms, 9th International Symposium, SEA 2010*, pages 350–360, 2010.
- [DFLL12] Claudia D’Ambrosio, Antonio Frangioni, Leo Liberti, and Andrea Lodi. A storm of feasibility pumps for nonconvex MINLP. *Math. Program.*, 136(2):375–402, 2012.
- [DMW16] Santanu S. Dey, Marco Molinaro, and Qianyi Wang. Analysis of sparse cutting-planes for sparse MILPs with applications to stochastic MILPs. <https://arxiv.org/abs/1601.00198>, 2016.
- [FGL05] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Math. Program.*, 104(1):91–104, 2005.

- [FL11] M. Fischetti and A. Lodi. Heuristics in mixed integer programming. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.
- [FS09] Matteo Fischetti and Domenico Salvagnin. Feasibility pump 2.0. *Math. Program. Comput.*, 1(2-3):201–222, 2009.
- [ILO] IBM ILOG. CPLEX high-performance mathematical programming engine. <http://www.ibm.com/software/integration/optimization/cplex/>.
- [KAA⁺11] Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, Emilie Danna, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Andrea Lodi, Hans D. Mittelmann, Ted K. Ralphs, Domenico Salvagnin, Daniel E. Steffy, and Kati Wolter. MIPLIB 2010. *Math. Program. Comput.*, 3(2):103–163, 2011.
- [MJPL92] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artif. Intell.*, 58(1-3):161–205, 1992.
- [MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, USA, 2005.
- [Pap91] Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *FOCS*. IEEE Computer Society, 1991.
- [Sch86] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.
- [Sch99] Uwe Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 410–414. IEEE Computer Society, 1999.
- [SLR10] Marianna De Santis, Stefano Lucidi, and Francesco Rinaldi. New concave penalty functions for improving the feasibility pump. Manuscript, 2010.

Appendix

A Minimal projected certificates can be found in polynomial time

Consider the following LP:

$$\begin{aligned} \max \quad & \lambda A \bar{x} - \lambda b \\ \text{s.t.} \quad & B^T \lambda = 0 \\ & e^T \lambda = 1 \\ & \lambda \geq 0, \end{aligned}$$

where e is the all-ones vector. Since we assumed a projected certificate exists, this LP is feasible and has strictly positive optimal value.

An optimal extreme point solution provides a projected certificate that can be computed in polynomial time [Sch86]; we just need to verify that there cannot exist a projected certificate with smaller support. Let λ^* be an extreme point optimal solution, and by contradiction assume that $\tilde{\lambda}$ gives a projected certificate and is such that $\text{supp}(\tilde{\lambda})$ is strictly contained in $\text{supp}(\lambda^*)$. Since $\tilde{\lambda} \geq 0$ and also different from 0, by scaling we can assume without loss of generality that $e^T \tilde{\lambda} = 1$, and thus $\tilde{\lambda}$ is a feasible solution for the LP above. This implies that

$$\begin{aligned} B^T (\lambda^* - \tilde{\lambda}) &= 0 \\ e^T (\lambda^* - \tilde{\lambda}) &= 0, \end{aligned}$$

so the assumption $\text{supp}(\tilde{\lambda}) \subsetneq \text{supp}(\lambda^*)$ implies that the columns of the matrix $\begin{bmatrix} B^T \\ e^T \end{bmatrix}$ in the support of λ^* are linearly dependent. But since λ^* is an extreme point, it is a basic solution, namely the columns of the matrix in the support of λ^* are linearly independent. This reaches a contradiction and concludes the proof.

B Original Feasibility Pump stalls even when flipping variables with zero fractionality is allowed

In Section 2 we showed that the original Feasibility Pump without restarts may stall; we now show that this is still the case even if variables with zero fractionality can be flipped in the perturbation step.

Let TT , the number of variables to be flipped, be randomly selected from the set $[t, T] \cap \mathbb{Z}$, where $T \in \mathbb{Z}_{++}$ is a pre-determined constant in the FP code (independent of the instance). Moreover assume the reasonable convention that for two variables with equal fractionality, we break ties using their index number, that is, if the x_i and x_j have the same fractionality and $i < j$, then x_i is picked before x_j to be flipped.

Consider the following subset-sum problem:

$$\begin{aligned} \max \quad & x_{T+2} \\ \text{s.t.} \quad & 5x_1 + \dots + 5x_{T+1} + 2x_{T+2} = 5T + 5 \\ & x_i \in \{0, 1\} \quad \forall i \in [T+2] \end{aligned}$$

Clearly the LP optimal solution \bar{x}^0 is of the form $\bar{x}_{T+2}^0 = 1$, $\bar{x}_i^0 = \frac{3}{5}$ for some $i \in [T+1]$ and $\bar{x}_j^0 = 1$ for all $j \in [T+1] \setminus \{i\}$. Rounding this we obtain \tilde{x}^0 which is of the form $\tilde{x}_{T+2}^0 = 1$ and $\tilde{x}_j^0 = 1$ for all $j \in [T+1]$. It is also straightforward to verify that \tilde{x}^0 is a stalling solution (see Definition 4.5). So that algorithm randomly selects TT from the set $[t, T] \cap \mathbb{Z}$ and flips TT variables. Note that only x_i has a fractionality of $|\frac{3}{5} - 1|$ and all the other variables have a fractionality of 0 for some $i \in [T+1]$. So using the convention for breaking ties, we flip x_i and $TT - 1$ other variables. Since $TT \leq T < T+1$, the new

point \tilde{x} is of the form $\tilde{x}_{T+2} = 1$ and $\tilde{x}_j = 0$ for $j \in S \subseteq [T+1]$ and $\tilde{x}_j = 1$ for $j \in [T+1] \setminus S$. (Note that S can also be \emptyset since we make no assumption on t).

First note that \tilde{x} is not a feasible solution since $\tilde{x}_{T+2} = 1$. Moreover,

1. If $S = \emptyset$, then $\tilde{x} = \tilde{x}^0$, a stalling solution visited before.
2. If $S \neq \emptyset$, then $5\tilde{x}_1 + \dots + 5\tilde{x}_{T+1} + 2\tilde{x}_{T+2} < 5T + 5$ and on projecting to the LP relaxation we will obtain a point of the form of \tilde{x}^0 . Rounding this again gives us \tilde{x}^0 , a stalling solution visited before.

This completes the proof.

C No long cycles in stalling

Lemma C.1. *Suppose that following is a sequence of points visited by Feasibility Pump (without any randomization):*

$$(\bar{x}^1, \bar{y}^1) \rightarrow (\tilde{x}^1, \bar{y}^1) \rightarrow (\bar{x}^2, \bar{y}^2) \rightarrow (\tilde{x}^2, \bar{y}^2),$$

where (\bar{x}^i, \bar{y}^i) , $i \in \{1, 2\}$ are the vertices of the LP relaxation, \tilde{x}^i , $i \in \{1, 2\}$ are 0-1 vectors, $\tilde{x}^i = \text{round}(\bar{x}^i)$ and $(\bar{x}^2, \bar{y}^2) = \ell_1\text{-proj}(\tilde{x}^1, \bar{y}^1)$. Then,

$$\|\bar{x}^1 - \tilde{x}^1\|_1 \geq \|\bar{x}^2 - \tilde{x}^2\|_1.$$

Proof. This result holds due to the fact that we are sequentially projecting using the same norm. In particular, we have that

$$\|\bar{x}^1 - \tilde{x}^1\|_1 \geq \|\bar{x}^2 - \tilde{x}^1\|_1,$$

since $(\bar{x}^2, \bar{y}^2) = \ell_1\text{-proj}(\tilde{x}^1, \bar{y}^1)$, i.e., \bar{x}^2 is a closest point in ℓ_1 -norm to \tilde{x}^1 in the projection of the LP relaxation in the x -space. Then

$$\|\bar{x}^2 - \tilde{x}^1\|_1 \geq \|\bar{x}^2 - \tilde{x}^2\|_1,$$

since \tilde{x}^1 and \tilde{x}^2 are both integer points and \tilde{x}^2 is obtained by rounding \bar{x}^2 (and a rounded point is the closest integer point in ℓ_1 norm). \square

A *long cycle* in feasibility pump is a sequence

$$(\bar{x}^1, \bar{y}^1) \rightarrow (\tilde{x}^1, \bar{y}^1) \rightarrow (\bar{x}^2, \bar{y}^2) \rightarrow (\tilde{x}^2, \bar{y}^2) \rightarrow \dots (\bar{x}^k, \bar{y}^k) \rightarrow (\tilde{x}^k, \bar{y}^k)$$

where

1. (\bar{x}^i, \bar{y}^i) , $i \in \{1, 2, \dots, k\}$ are the vertices of the LP relaxation, \tilde{x}^i , $i \in \{1, 2, \dots, k\}$ are 0-1 vectors, $\tilde{x}^i = \text{round}(\bar{x}^i)$ and $(\bar{x}^{i+1}, \bar{y}^{i+1}) = \ell_1\text{-proj}(\tilde{x}^i, \bar{y}^i)$,
2. $\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^{k-1}$ are unique integer vectors,
3. $\bar{x}^1 = \bar{x}^k$, $\tilde{x}^1 = \tilde{x}^k$, and
4. $k \geq 3$.

The statement of Theorem 4.1 is that such a scenario cannot occur, assuming 0.5 is always rounded consistently.

Proof of Theorem 4.1. Without loss of generality, we assume that 0.5 is rounded up to 1. Consider the sub-sequence $(\bar{x}^i, \bar{y}^i) \rightarrow (\tilde{x}^i, \bar{y}^i) \rightarrow (\bar{x}^{i+1}, \bar{y}^{i+1}) \rightarrow (\tilde{x}^{i+1}, \bar{y}^{i+1})$. By Lemma C.1, since there is cycling, we have that

$$\|\bar{x}^i - \tilde{x}^i\|_1 = \|\bar{x}^{i+1} - \tilde{x}^i\|_1 = \|\bar{x}^{i+1} - \tilde{x}^{i+1}\|_1.$$

For simplicity and without loss of generality, we may assume that \tilde{x}^i is the all ones vector. (This can be achieved by reflecting on coordinates the LP relaxation and the $[0, 1]^n$ hypercube. Note that under such mappings, the sequence of points in feasibility pump will not be altered. Moreover, a point with value 0.5 in some coordinates $I \subseteq [n]$ will be mapped to a point with 0.5 in the coordinates I .)

Let $\emptyset \neq J \subseteq [n]$ be the set of indices where $\tilde{x}_j^i \neq \tilde{x}_j^{i+1}$, that is $\tilde{x}_j^{i+1} = 0$ for all $j \in J$. Since $\|\bar{x}^{i+1} - \tilde{x}^i\|_1 = \|\bar{x}^{i+1} - \tilde{x}^{i+1}\|_1$, we have

$$\begin{aligned} \sum_{j=1}^n (1 - \bar{x}_j^{i+1}) &= \sum_{j \in [n] \setminus J} (1 - \bar{x}_j^{i+1}) + \sum_{j \in J} \bar{x}_j^{i+1} \\ \Leftrightarrow \sum_{j \in J} \bar{x}_j^{i+1} &= \frac{|J|}{2}. \end{aligned} \tag{5}$$

Now observe that since $\tilde{x}_j^{i+1} = 0$ for $j \in J$, we must have that $\bar{x}_j^{i+1} < 0.5$ for all $j \in J$. This contradicts, (5). □